Application No: unknown
Applicant: AMD
Our ref: P33673-03486
Date: September 16, 2002

CLAIMS

1. A SMBus message handler comprising:

   a memory (202) configured to store microcode comprising at least two programs (210, 211, 212) each for handling a bus command protocol and comprising at least one instruction;

   an interface to a register (SMB_PRTCL) configured to identify a starting address of a program in said memory;

   an instruction fetch unit (203) configured to read an instruction at an address in said memory (202); said address being specified by a program counter (pc); and

   a finite-state machine (201) configured to receive and interpreting the instructions read by said instruction fetch unit (203) and for managing the data transfer between an SMBus (213, 214) interface, and a register set (208) in compliance with said instructions read from said memory.

2. The SMBus message handler of claim 1, wherein said register set complies with the ACPI specification.

3. The SMBus message handler of claim 1, further comprising an address register array (207) comprising a plurality of starting addresses of programs stored in said memory, said register (SMB_PRTCL) comprising an offset for pointing at a specific register in said address register array.

4. The SMBus message handler of claim 2, further comprising a buffer pointer register (206) for pointing at one of a plurality of data registers (SMB_DATA); said finite-state machine (201) transferring data read from the SMBus (213,

214) interface to the data register at which said buffer pointer register (206) points if said finite-state machine (201) interprets a "receive data to" instruction; said finite-state machine (201) transferring the data read from the data register at which said buffer pointer register points to said SMBus (213, 214) interface if said finite-state machine (201) interprets a "transmits data from" instruction.

5. The SMBus message handler of claim 4, wherein the finite-state machine causes said buffer pointer register (206) to be incremented each time a "transmit data to" or a "transmit data from" instruction is executed.

6. The SMBus message handler of claim 1, further comprising a loop counter (204) for storing the value of a block counter register SMB_BCNT in said loop counter (204) if the finite-state machine executes a "transmit data from SMB_BCNT" instruction; said loop counter (204) being decremented each time a data byte is transmitted to said SMBus (213, 214) interface while a "transmit data from" instruction is executed and the "transmit data from" instruction be completed when the value of said loop counter (204) reaches zero.

7. The SMBus message handler of claim 1, further comprising a loop counter (204) and a block count register (SMB_BCNT) both for storing a byte received from said SMBus (213, 214) interface if the finite-state machine (201) executes a "receive data to SMB_BCNT" instruction, said loop counter (204) being decremented each time a data byte is transmitted to or received from said SMBus (213, 214) interface while a "received data to" instruction is executed and the "received data to" instruction being completed when the value of said loop counter (204) reaches zero.

8. The SMBus message handler of claim 1, wherein each instruction comprises one bit (304) indicating as to whether or not an instruction is the last instruction in the program.

9. The SMBus message handler of claim 1, wherein each instruction comprises one bit (305) indicating as to whether an instruction is to be executed only once or this instruction is to be executed repeatedly until a loop counter (204) becomes zero, wherein said loop counter is decremented each time an instruction is executed repeatedly.

10. An integrated circuit chip for transmitting and receiving data over a SMBus comprising:

    an interface to a memory (202) configured to store microcode comprising at least two programs (210, 211, 212) each for handling a bus command protocol and comprising at least one instruction;

    an interface to a register (SMB_PRTCL) configured to identify a starting address of a program in said memory;

    an instruction fetch unit (203) configured to read an instruction at an address in said memory (202); said address being specified by a program counter (pc); and

    a finite-state machine (201) configured to receive and interpret the instructions read by said instruction fetch unit (203) and for managing the data transfer between an SMBus (213, 214) interface, and a register set (208) in compliance with said instructions read from said memory.

11. The integrated circuit chip of claim 10, wherein said register set complies with the ACPI specification.

12. The integrated circuit chip of claim 10, further comprising an address register array (207) comprising a plurality of starting addresses of programs stored in said memory, said register (SMB_PRTCL) comprising an offset for pointing at a specific register in said address register array.

13. The integrated circuit chip of claim 11, further comprising a buffer pointer register (206) for pointing at one of a plurality of data registers (SMB_DATA) comprised in register set (208); said finite-state machine (201) transferring data read from the SMBus (213, 214) interface to the data register at which said buffer pointer register (206) points if said finite-state machine (201) interprets a "receive data to" instruction; said finite-state machine (201) transferring the data read from the data register at which said buffer pointer register points to said SMBus (213, 214) interface if said finite-state machine (201) interprets a "transmits data from" instruction.

14. The integrated circuit chip of claim 13, wherein the finite-state machine (201) causes said buffer pointer register (206) to be incremented each time a "transmit data to" or a "transmit data from" instruction is executed.

15. The integrated circuit chip of claim 10, further comprising a loop counter (204) for storing the value of a block counter register SMB_BCNT in said loop counter (204) if the finite-state machine executes a "transmit data from SMB_BCNT" instruction; said loop counter (204) being decremented each time a data byte is transmitted to said SMBus (213, 214) interface while a "transmit data from" instruction is executed and the "transmit data from" instruction be completed when the value said loop counter (204) reaches zero.

16. The integrated circuit chip of claim 10, further comprising a loop counter (204) and a block count register (SMB_BCNT) comprised in said register set (208) both for storing a byte received from said SMBus (213, 214) interface if the finite-state machine (201) executes a "receive data to SMB_BCNT" instruction, said loop counter (204) being decremented each time a data byte is transmitted to or received from said SMBus (213, 214) interface while a "received data to" instruction is executed and the "received data to" instruction being completed when the value of said loop counter (204) reaches zero.

17. The integrated circuit chip of claim 10, wherein each instruction comprises one bit (304) indicating as to whether or not an instruction is the last instruction in the program.

18. The integrated circuit chip of claim 10, wherein each instruction comprises one bit (305) indicating as to whether an instruction is to be executed only once or this instruction is to be executed repeatedly until a loop counter (204) becomes zero, wherein said loop counter is decremented each time an instruction is executed repeatedly.

19. Method for controlling an SMBus:

identifying (402) a starting address of a program (210, 211,212) comprising one or more instructions; said program (210, 211, 212) being stored in a memory (202);

fetching instructions of said program one after another into a finite-state machine (201); and

transferring data between an SMBus (213, 214) interface and a register set in compliance with the instruction present in said finite-state machine (201).

20. Method of claim 19, wherein said register set complies with the ACPI specification.

21. Method of claim 19, wherein said identifying step comprises the sub-steps of:

    reading a first value of a protocol register (SMB_PRTCL) specifying an offset in an address register array (207);

    reading a second value of a register of said address register array (207), said register being specified by said offset; said second value constituting said starting address of said program.

22. Method of claim 20, wherein said transferring step comprising the sub-steps of:

    interpreting a "received data to" instruction; reading the value of a buffer pointer register (206); and transferring the data read from said SMBus (213, 214) interface to the data register (SMB_DATA) at which the value stored in said buffer pointer register (206) points.

23. Method of claim 22, wherein said transferring step further comprises incrementing said value of said buffer pointer register (206).

24. Method of claim 22, wherein said transferring step further comprising decrementing a loop counter (204) and checking as to whether said loop counter (204) has a value of zero.

25. Method of claim 20, wherein said transferring step comprises the sub-steps of:

    interpreting a "transmit data from" instruction;

    reading the value of a buffer pointer register (206); and transferring the data read from the data register (SMB_DATA) at which the value stored in said buffer pointer register (206) to said SMBus (213, 214) interface.

26. Method of claim 25, wherein said transferring step further comprises incrementing of said buffer pointer register (206).

27. The method of claim 25, wherein the transferring step further comprises decrementing of said loop counter (204).

28. The method of claim 19, wherein said transferring step comprises:

interpreting a "transmit data from SMB_BCNT" instruction;

storing the value of a block count register SMB BCNT in a loop counter (204); and

transmitting the value of said block count register (SMB_BCNT) to said SMBus (213, 214) interface.

29. The method of claim 19, wherein said transferring step comprises:

interpreting a "received data to SMB_BCNT" instruction;

transmitting a byte from said SMBus (213, 214) interface to a block count register SMB_BCNT; and

storing the value of the byte received from said SMBus (213, 214) interface to a loop counter register (204).

30. The method of claim 19, wherein the transferring step further comprises:

determining as to whether a stop bit (304) has a predetermined value; if this is the case:

writing 80h into a status register (SMB_STS) of said register set (208).

31. The method of claim 19, wherein the transferring step further comprises:

determining as to whether a loop bit (305) of an instruction has a predetermined value; if that is the case,

executing said instruction repeatedly;

decrementing a loop counter (204) each time said instruction is executed;

finishing the execution of said loop instruction when the value of the said loop counter (204) becomes zero; and

fetching the next instruction.

32. A SMBus test device comprising:

a memory configured to store sequences of instructions (604, 606, 607, 609, 611, 612, 614, 616, 670, 618, 620, 621, 623, 625, 626, 628, 630, 631, 633, 635, 636, 638, 640, 642, 644, 646 648, 650, 652, 654, 656, 658 to 669) and

a SMBus interface (505) to which a plurality of SMBus devices (521, 522, 523) can be connected;

an interface (RS-232) configured to input keys;

a processor performing the following;

checking as to whether a key is input;

upon inputting of a key checking (603, 605, 608, 610, 613, 615, 617, 619, 622, 624, 627, 629, 632, 634, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657) as to whether said key can be mapped to a sequence of instructions (604, 606, 607, 609, 611, 612, 614, 616, 670, 618, 620, 621, 623, 625, 626, 628, 630, 631, 633, 635, 636, 638, 640, 642, 644, 646 648, 650, 652, 654, 656, 658 to 669) for controlling devices connected to said SMBus interface or transferring data to or receiving data from said devices connected to said SMBus; and

executing said sequence of instructions (604, 606, 607, 609, 611, 612, 614, 616, 670, 618, 620, 621, 623, 625, 626, 628, 630, 631, 633, 635, 636, 638, 640, 642, 644, 646 648, 650, 652, 654, 656, 658 to 669) to which said key has been mapped.

33. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “0” (603) and if that is the case performing a block write I^2C operation (604).

34. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "1" (605) and if that is the case performing a block read I^2C operation (606).

35. The SMBus test device of claim 34, wherein the processor turns the PEC on or off (607).

36. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "2" (608) and if that is the case performs a quick write protocol (609).

37. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "3" (610) and if that is the case performing a quick read protocol (611).

38. The SMBus test device of claim 37, wherein the processor turns the PEC on or off (612).

39. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "4" and if that is the case performs a send byte protocol (614).

40. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "5" (615) and if that is the case performs a receive byte protocol (616).

41. The SMBus test device of claim 40, wherein the processor turns the PEC on or off (670).

42. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "6" (617) and if that is the case performs a read byte protocol (618).

43. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "7" (619) and if that is the case performs a read byte protocol (620).

44. The SMBus test device of claim 43, wherein the processor turns the PEC on or off (621).

45. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “8” (622) and if that is the case performs a write word protocol (623).

46. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “9” (624) and if that is the case performs a read word protocol (625).

47. The SMBus test device of claim 46, wherein the processor turns the PEC on or off (626).

48. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “A” (627) and if that is the case performs a write block protocol (628).

49. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “B” (629) and if that is the case performs a read block protocol (630).

50. The SMBus test device of claim 49, wherein the processor turns the PEC on or off (631).

51. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “C” (632) and if that is the case performs a process called protocol (633).

52. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “D” (634) and if that is the case performs a block write-block read process call (635).

53. The SMBus test device of claim 52, wherein the processor further turns the PEC on or off (636).

54. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “E” (637) and if that is the case notifies the host of an alarm (638).

55. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "F" (639) and if that is the case the SMBus test device sends a remote control message (640).

56. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "S" (641) and if that is the case inputs two hexadecimal characters and saves them as the new device slave address.

57. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "P" (643) and if that is the case turns the PEC on or off (644).

58. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "V" (645) and if that is the case turns the verbose mode on or off (646); in verbose mode each byte transmitted to or received from the SMBus interface (505) is transmitted to the second interface (RS-232).

59. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "M" (647) and if that is the case the processor receives (701, 705) two hexadecimal characters from the second interface constituting an address within the auxiliary RAM of the test device; the processor further receiving (708, 712) two hexadecimal characters from said second interface constituting the new value of said byte in said auxiliary RAM.

60. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "N" (649) and if that is the case the processor receives two numeric characters (721, 725) from the second interface; the processor will generate a ACK/NACK at a position in a SMBus protocol at the position specified by said two numeric characters.

61. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "O" (651) and if that is the case the processor inputs (741, 745) two hexadecimal characters from said second interface (RS-232) and adds a byte specified by said two characters to the PEC byte.

62. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is "T" (653) and if that is the case the processor inputs

(761, 765) two numeric keys from said second interface specifying the location for an error; the processor further inputting (769, 773) two other numeric keys specifying a timeout time.

63. The SMBus test device of claim 32, wherein the processor checks as to whether the input key is “G” (657) and if that is the case the processor inputs further keys from said second interface and depending on the keys inputted the SMBus test device sends a reset status message (660), a power down message (662), a power up message (664), a power cycle message (666) to said SMBus interface (505).

64. A method for controlling a testing system (500) for SMBus chip set validation comprising:

    checking (602) as to whether a key is input from a second interface;

    upon inputting said key checking (603, 605, 608, 610, 613, 615, 617, 619, 622, 624, 627, 629, 632, 634, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657) as to whether said key can be mapped to a sequence of instructions for controlling devices connected to a SMBus or transferring data to or receiving data from said devices connected to said SMBus; and

    executing (604, 606, 607, 609, 611, 612, 614, 616, 670, 618, 620, 621, 623, 625, 626, 628, 630, 631, 633, 635, 636, 638, 640, 642, 644, 646 648, 650, 652, 654, 656, 658 to 669) said sequence of instructions to which said key has been mapped.

65. The method of claim 64, further comprising checking as to whether the input key is “0”(603) and if that is the case performing a block write I^2C operation (604).

66. The method of claim 64, further comprising checking as to whether the input key is “1” (605) and if that is the case performing a block read I^2C operation (606).

67. The SMBus test device of claim 66, wherein the processor turns the PEC on or off (607).

68. The method of claim 64, further comprising checking as to whether the input key is “2” (608) and if that is the case performs a quick write protocol (609).

69. The method of claim 64, further comprising checking as to whether the input key is “3” (610) and if that is the case performing a quick read protocol (611).

70. The method of claim 69, further comprising turning the PEC on or off (612).

71. The method of claim 64, further comprising checking as to whether the input key is "4” and if that is the case performs a send byte protocol (614).

72. The method of claim 64, further comprising checking as to whether the input key is “5” (615) and if that is the case performs a receive byte protocol (616).

73. The method of claim 72, further comprising turning the PEC on or off (670).

74. The method of claim 64, further comprising checking as to whether the input key is “6” (617) and if that is the case performs a read byte protocol (618).

75. The method of claim 64, further comprising checking as to whether the input key is “7” (619) and if that is the case performs a read byte protocol (620).

76. The method of claim 75, further comprising turning the PEC on or off (621).

77. The method of claim 64, further comprising checking as to whether the input key is “8” (622) and if that is the case performs a write word protocol (623).

78. The method of claim 64, further comprising checking as to whether the input key is “9” (624) and if that is the case performs a read word protocol (625).

79. The method of claim 78, further comprising turning the PEC on or off (626).

80. The method of claim 64, further comprising checking as to whether the input key is “A” (627) and if that is the case performs a write block protocol (628).

81. The method of claim 64, further comprising checking as to whether the input key is “B” (629) and if that is the case performs a read block protocol (630).

82. The method of claim 81, further comprising turning PEC on or off (631).

83. The method of claim 64, further comprising checking as to whether the input key is "C" (632) and if that is the case performs a process called protocol (633).

84. The method of claim 64, further comprising checking as to whether the input key is "D" (634) and if that is the case performs a block write-block read process call (635).

85. The method of claim 84, further comprising turning the PEC on or off (636).

86. The method of claim 64, further comprising checking as to whether the input key is "E" (637) and if that is the case notifies the host of an alarm (638).

87. The method of claim 64, further comprising checking as to whether the input key is "F" (639) and if that is the case the SMBus test device sends a remote control message (640).

88. The method of claim 64, further comprising checking as to whether the input key is "S" (641) and if that is the case inputs two hexadecimal characters and saves them as the new device slave address.

89. The method of claim 64, further comprising checking as to whether the input key is "P" (643) and if that is the case turns the PEC on or off (644).

90. The method of claim 64, further comprising checking as to whether the input key is "V" and if that is the case turns the verbose mode on or off (646); in verbose mode each byte transmitted to or received from the SMBus interface (505) is transmitted to the second interface (RS-232).

91. The method of claim 64, further comprising checking as to whether the input key is "M" (647) and if that is the case the processor receives (701, 705) two hexadecimal characters from the second interface constituting an address within the auxiliary RAM of the test device; the processor further receiving (708, 712) two hexadecimal characters from said second interface constituting the new value of said byte in said auxiliary RAM.

92. The method of claim 64, further comprising checking as to whether the input key is "N" (649) and if that is the case the processor receives two numeric characters (721, 725) from the second interface; the processor will generate a

ACK/NACK at a position in a SMBus protocol at the position specified by said two numeric characters.

93. The method of claim 64, further comprising checking as to whether the input key is "O" (651) and if that is the case the processor inputs (741, 745) two hexadecimal characters from said second interface (RS-232) and adds a byte specified by said two characters to the PEC byte.

94. The method of claim 64, further comprising checking checks as to whether the input key is "T" (653) and if that is the case the processor inputs (761, 765) two numeric keys from said second interface specifying the location for an error; the processor further inputting (769, 773) two other numeric keys specifying a timeout time.

95. The method of claim 64, further comprising checking as to whether the input key is "G" (657) and if that is the case the processor inputs further keys from said second interface and depending on the keys inputted the SMBus test device sends a reset status message (660), a power down message (662), a power up message (664), a power cycle message (666) to said SMBus interface (505).